Math

9

ALTERNATIVE METHODS FOR SOLVING THE CC-TABLE IN GIMPEL'S ALGORITHM FOR SYNTHESIZING OPTIMAL THREE LEVEL NAND NETWORKS

by

Keith R. Hohulin
Saburo Muroga

April, 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-75-720

ALTERNATIVE METHODS FOR SOLVING THE CC-TABLE IN GIMPEL'S ALGORITHM FOR SYNTHESIZING OPTIMAL THREE LEVEL NAND NETWORKS

by

Keith R. Hohulin
Saburo Muroga

April, 1975

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A classic paper by J.F. Gimpel entitled "The Minimization of TANT Networks" [1] details the theory of an algorithm which synthesizes optimal TANT (three level AND-NOT (NAND)) networks with a minimum number of gates (the number of connections is not necessarily minimized) under the assumption that only the uncomplemented switching variables are available as inputs to the networks. The most time-consuming calculation in Gimpel's algorithm, which is an important contribution to NAND (or NOR) network design, is obtaining the solution to the CC-table (defined in Chapter 2). When the number of variables of a switching function increases, the computation time for this portion determines the applicability of Gimpel's algorithm. (A similar difficulty occurs when a minimal sum for a switching function is to be obtained from a prime implicant table by Petrick's method (e.g., see [3]). Multiplying out Petrick's function is very time-consuming, though it appears simple at first glance.) Because the CC-table can be solved in a number of ways, Gimpel's algorithm can be implemented in computer programs in many different ways.

A brief description of the CC-table and the straightforward algebraic approach for finding its solutions is given in Chapter 2. Chapter 3 describes several alternative methods of solution for the cc-table and provides an example which illustrates the differences among these methods. Finally, Chapter 4 contains the results of implementing the various methods into a computer program and running some test functions on the IBM 360/75 at the University of Illinois.

CHAPTER 2

THE CC-TABLE

In his paper [1] Gimpel presents a series of definitions and theorems which lead, ultimately, to the formation of a cover and closure table (CC-table) for a given switching function, f. In general a CC-table is a rectangular matrix of dots, ×'s and blanks such that each column has at least one × and at most one dot. Columns with no dots are called cover columns while columns with a dot are called closure columns. A row having an × entry in a column is said to cover that column and a row with a dot in a column is said to imply that column. A set of rows Z is said to be a cover if every cover column is covered by some row of Z. A set of rows Z is said to be closed if every column implied by a row of Z is covered by some other row of Z. A closed cover with the minimum number of rows in a CC-table yields a solution to the minimization problem of TANT networks.

Fig. 2.2 shows the CC-table for the function represented by the Veitch map in Fig. 2.1. This table is actually a reduced form of the original CC-table for this function. That is, an initial CC-table was formed and then several reduction techniques developed by Gimpel were applied to it to eliminate some rows and/or columns. The arrangement of this table allows a distinction to be made between "left columns" and "right columns" as well as "upper rows" and "lower rows". In Fig. 2.2 columns (1)–(6) and (7)–(17) are the left and right columns, respectively,

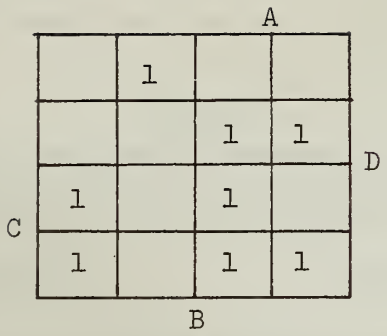

Fig. 2.1. The Veitch map for a function used to illustrate the CC-table.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)
a	×			×			•	•									
b				×					•	◦							
c	×										•	•					
d				×	×								•				
e					×	×											
f						×											
g			×											•			
h		×													◦	◦	•
i											×				×		
j														×		×	
k									×				×				×
ℓ							×						×	×			
m								×		×		×				×	
n									×				×	×			
o											×			×			

Fig. 2.2 A reduced CC-table for the function in Fig. 2.1.

while rows a – h and i – o are the upper and lower rows, respectively. The solutions of the CC-table in Fig. 2.2 are the minimal TANT networks for the function in Fig. 2.1.

The straightforward algebraic method for solving a CC-table is summarized as follows [2]: for each column (u) of the table, a switching equation is written in the form

$$\bar{\gamma}_0 \vee \gamma_1 \vee \cdots \vee \gamma_y = 1 \qquad \text{(A)}$$

where γ_0 is the variable associated with the row that has a dot in column u (if column (u) does not contain a dot, the term $\bar{\gamma}_0$ is not contained in (A)) and $\gamma_1, \ldots, \gamma_y$ are the variables corresponding to the rows with X's in column u. For example, the equation for column (11) in Fig. 2.2 is $\bar{c} \vee i \vee 0 = 1$. It is convenient to use the notation (j) to refer to both the j th column of the CC-table and the alterm corresponding to that column (i.e., (11) will be used to represent alterm $(\bar{c} \vee i \vee 0)$). The requirements of the entire CC-table can be represented in a single switching equation by forming the logical product of the equations of the form (A) for all of the columns in the CC-table ((1)(2)(3)...(17) for the CC-table in Fig. 2.2.). This single equation is multiplied out into an irredundant disjunctive form. The solutions to the table are the terms of the irredundant disjunctive form which contain the smallest number of uncomplemented variables.

CHAPTER 3

ALTERNATIVE METHODS FOR SOLVING THE CC-TABLE

3.1. Ordering the alterms.

The most straightforward approach for solving the CC-table is to multiply out the alterms in the same order as they are formed from the CC-table. For the example in Fig. 2.2 this order is simply (1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12)(13)(14)(15)(16)(17). First, alterms (1) and (2) are multiplied together and the result can be represented by ((1)(2)). Next, ((1)(2)) is multiplied by (3) to get (((1)(2))(3)). This process is repeated until all of the alterms have been multiplied together. For functions with a large number, say M, of alterms, the number of terms in the j th partial product can become quite large as j approaches M where the j th partial product is the result of multiplying out the first j alterms.

To some extent the large size of the partial products is unavoidable. However, the above ordering of alterms does not take advantage of the fact that multiplying two alterms which have some common literals will result in a smaller number of terms than multiplying two alterms with the same number of literals as before but with no common literals. That is, by rearranging the order of the alterms so that alterms with common literals are adjacent, the number of terms in the partial product may be greatly reduced. Two such orderings were tried. In both cases the alterms corresponding to the left columns of the cc-table are ordered separately from the alterms corresponding to the right columns. Also, only the uncomplemented literals

in the alterms (i.e., the ×'s in the table) are considered. The two orderings are:

ORDERING 1: Assume that k alterms have been ordered (initially $k = 1$). Search the alterms to the right of (k) and select all the alterms (say (j_1), (j_2), ..., (j_p) where $j_1 < j_2 < \ldots < j_p$) which contain at least one literal in common with (k). Then interchange alterms (k+1) and (j_1), (k+2) and (j_2), ..., and (k+p) and (j_p). Of course, if there are no such (j_i), then p is zero and no interchanges are made. Set k to $b+p+1$ and repeat until all of the alterms are ordered. The result of applying this ordering to the table in Fig. 2 is as follows:

(1)(4)(3)(2)(5)(6)(7)(13)(14)(10)(12)(8)(16)(9)(17)(11)(15).

ORDERING 2: Assume that k alterms have been ordered (initially $k = 0$). From the alterms to the right of (k), choose the alterm with the largest number of literals and interchange it with alterm (k+1). Then, from the alterms to the right of (k+1), choose the alterm which has the largest number of literals in common with alterm (k+1) and interchange it with alterm (k+2). If no alterm has any common literals with alterm (k+1), then no interchange is made with alterm (k+2). Set k to $k+2$ and repeat until the ordering is complete. Appling ORDERING 2 to the table in Fig. 2 gives (4)(1)(5)(6)(3)(2)(14)(13)(9)(17)(11)(15)(16)(12)(7)(8)(10).

3.2. Methods 1 through 5.

The CC-table in Fig. 3.2.1 will be used as an example to illustrate the calculation done by each of the methods discussed in this chapter. For each method the terms representing the solutions are underlined in the final expression.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
a	×			×				
b		×			·			
c		×	×	×		·	·	
d	×		×					
e		×	×					·
f					×		×	×
g						×		×

Fig. 3.2.1 The CC-table used in the example.

Each of the methods in this section uses a different approach for multiplying out the products of sums formed from the CC-table. However, all of the methods in this section multiply together all of the alterms to get the desired irredundant disjunctive form. A completely different approach is presented in Section 3.3.

METHOD 1: This is the straightforward method of calculation which was discussed at the beginning of Section 3.1. This method yields the following calculation for the CC-table in Fig. 3.2.1.

$$(1)(2)(3)(4)(5)(6)(7)(8)$$

$$= (a \vee d)(b \vee c \vee e)(c \vee d \vee e)(a \vee c)(\bar{b} \vee f)(\bar{c} \vee g)(\bar{c} \vee f)(\bar{e} \vee f \vee g)$$

$$= (ab \vee bd \vee ac \vee cd \vee ae \vee de)(c \vee d \vee e) \ \ldots \ (\bar{e} \vee f \vee g)$$

$$= (ac \vee cd \vee bd \vee de \vee ae)(a \vee c) \ \ldots \ (\bar{e} \vee f \vee g)$$

$$= (ac \vee abd \vee ae \vee cd)(\bar{b} \vee f) \ \ldots \ (\bar{e} \vee f \vee g)$$

$$= (a\bar{b}c \vee a\bar{b}e \vee \bar{b}cd \vee acf \vee abdf \vee aef \vee cdf)(\bar{c} \vee g) \ \ldots \ (\bar{e} \vee f \vee g)$$

$$= (a\overline{bc}e \vee ab\bar{c}df \vee a\bar{c}ef \vee a\bar{b}cg \vee a\bar{b}eg \vee \bar{b}cdg \vee acfg \vee abdfg \vee aefg \vee cdfg)$$
$$(\bar{c} \vee f)(\bar{e} \vee f \vee g)$$

$$= (a\overline{bc}e \vee ab\bar{c}df \vee a\bar{c}ef \vee acfg \vee abdfg \vee aefg \vee cdfg)(\bar{e} \vee f \vee g)$$

$$= (ab\bar{c}df \vee \underline{a\bar{c}ef} \vee acfg \vee abdfg \vee aefg \vee cdfg \vee \underline{a\overline{bc}eg})$$

METHOD 2: ORDERING 1 is applied to both the left alterms (the alterms formed from the left columns) and the right alterms (the alterms formed from the right columns). The alterms are then multiplied out in exactly the same way as in METHOD 1. Applying METHOD 2 to the CC-table of Fig. 3.2.1 yields the following calculation:

$$(1)(3)(4)(2)(5)(7)(8)(6)$$

$$= (a \vee d)(c \vee d \vee e)(a \vee c)(b \vee c \vee e)(\bar{b} \vee f)(\bar{c} \vee f)(\bar{e} \vee f \vee g)(\bar{c} \vee g)$$

$$= (ac \vee d \vee ae)(a \vee c) \ \ldots \ (\bar{c} \vee g)$$

$$= (ac \vee ad \vee ae \vee cd)(b \vee c \vee e) \ \ldots \ (\bar{c} \vee g)$$

$$= (abd \vee ac \vee cd \vee ae)(\bar{b} \vee f) \ \ldots \ (\bar{c} \vee g)$$

$$= (a\bar{b}c \vee \bar{b}cd \vee a\bar{b}e \vee abdf \vee acf \vee cdf \vee aef)(\bar{c} \vee f) \ \ldots \ (\bar{c} \vee g)$$

$$= (a\overline{bc}e \vee abdf \vee acf \vee cdf \vee aef)(\bar{e} \vee f \vee g)\ (\bar{c} \vee g)$$

$$= (abdf \vee acf \vee cdf \vee aef \vee a\overline{bc}eg)(\bar{c} \vee g)$$

$$= (ab\bar{c}df \vee \underline{a\bar{c}ef} \vee \underline{a\overline{bc}eg} \vee abdfg \vee acfg \vee cdfg \vee aefg)$$

METHOD 3: ORDERING 1 is applied to both the left and right alterms. The alterms are then multiplied in pairs to get a set of partial products. These partial products are then multiplied out in order from left to right to obtain the solutions.

METHOD 3': The CC-table is solved in exactly the same way as in METHOD 3. However, two other changes were made to the program which was used to implement METHODS 1, 2 and 3. First, when two alterms (i) and (j) are multiplied together to form the partial product ((i)(j)), the $a \vee ab = a$ rule may allow some terms to be eliminated from the partial product. The program for METHODS 1, 2 and 3 always checks a newly formed partial product to see if any terms can be eliminated. However, it is clear that if alterms (i) and (j) have no common literals, then no terms can be removed from the partial product ((i)(j)). The program was changed so that a new partial product is not checked to see if any terms can be eliminated if the alterms (or partial products) which formed the new partial product have no common literals. Second, the procedure for removing the terms which can be eliminated from a partial product was changed.

The two changes to the program discussed in the description of METHOD 3' are included in METHODS 4 and 5 of this section as well as METHODS 6, 7 and 8 of Section 3.3. The calculation below shows the result of applying METHODS 3 and 3' to Fig. 3.2.1.

$$(1)(3)(4)(2)(5)(7)(8)(6)$$

$$= (a \vee d)(c \vee d \vee e)(a \vee c)(b \vee c \vee e)(\overline{b} \vee f)(\overline{c} \vee f)(\overline{e} \vee f \vee g)(\overline{c} \vee g)$$

$$= (ac \vee d \vee ae)(ab \vee c \vee ae)(\overline{bc} \vee f)(\overline{ce} \vee \overline{c}f \vee g)$$

$$= (abd \vee ac \vee cd \vee ae)(\overline{bc} \vee f)(\overline{ce} \vee \overline{c}f \vee g)$$

$$= (a\overline{bc}e \vee abdf \vee acf \vee cdf \vee aef)(\overline{ce} \vee \overline{c}f \vee g)$$

$$= (ab\overline{c}df \vee \underline{\overline{a}cef} \vee \underline{a\overline{bc}eg} \vee abdfg \vee acfg \vee cdfg \vee aefg)$$

METHOD 4: ORDERING 1 is applied to both the left and right alterms. From the description of ORDERING 1, it is clear that the alterms will be rearranged into groups of one or more alterms which have at least one literal in common with the leftmost alterm of the group. The alterms in each group are multiplied together to get a set of partial products. The partial products are then multiplied together in order from left to right. METHOD 4 applied to Fig. 3.2.1 yields

$$(1)(3)(4)(2)(5)(7)(8)(6)$$

$$= (a \vee d)(c \vee d \vee e)(a \vee c)(b \vee c \vee e)(\overline{b} \vee f)(\overline{c} \vee f)(\overline{e} \vee f \vee g)(\overline{c} \vee g)$$

$$= (ac \vee ad \vee ae \vee cd)(b \vee c \vee e)(\overline{bce} \vee f \vee \overline{bc}g)(\overline{c} \vee g)$$

$$= (abd \vee ac \vee cd \vee ae)(\overline{bce} \vee f \vee \overline{bc}g)(\overline{c} \vee g)$$

$$= (abdf \vee acf \vee cdf \vee aef \vee a\overline{bc}eg)(\overline{c} \vee g)$$

$$= (ab\overline{c}df \vee \underline{\overline{a}cef} \vee \underline{a\overline{bc}eg} \vee abdfg \vee acfg \vee cdfg \vee aefg)$$

METHOD 5: ORDERING 1 is applied to the left and right alterms. The alterms are then multiplied in pairs to get a set of partial products. These partial products are also multiplied in pairs to get another set of partial products. This process is continued until at the last step the final two partial products are multiplied together to obtain the solution. In other words, pairwise multiplication is used throughout the calculation. METHOD 5 applied to Fig. 3.2.1 yields

$$(1)(3)(4)(2)(5)(7)(8)(6)$$

$$= (a \vee d)(c \vee d \vee e)(a \vee c)(b \vee c \vee e)(\bar{b} \vee f)(\bar{c} \vee f)(\bar{e} \vee f \vee g)(\bar{c} \vee g)$$

$$= (ac \vee d \vee ae)(ab \vee c \vee ae)(\overline{bc} \vee f)(\overline{ce} \vee \bar{c}f \vee g)$$

$$= (ac \vee abd \vee cd \vee ae)(\overline{bce} \vee \bar{c}f \vee \overline{bc}g \vee fg)$$

$$= (ab\bar{c}df \vee \underline{a\bar{c}ef} \vee \underline{ab\overline{ce}g} \vee acfg \vee abdfg \vee cdfg \vee aefg)$$

3.3. Methods 6 through 8.

The organization of the CC-table for the minimization problem of TANT networks which allows a distinction between left and right columns as well as upper and lower rows (as discussed in Chapter 2) suggests an alternative approach for solving the CC-table. In this approach the sums of literals formed from the left columns are multiplied together to get a partial product. For each term α in the partial product, only the alterms formed from the right columns which have dots in rows which are represented by literals in α need to be multiplied out and the resulting terms concatenated with α to form closed covers for the CC-table. The complemented literals which represent the dots are not included in these alterms since the terms they would be in when concatenated with α would result in a term of the form $x\bar{x}\beta$. Following this procedure for each α will yield all of the closed covers for the CC-table.

As an example, consider Table 2.2. Multiplying together the alterms formed from the left columns $((a \vee c)(h)(g)(a \vee b \vee d)(d \vee e)(e \vee f))$ results in the partial product $aegh \vee adfgh \vee bcegh \vee cdegh \vee cdfgh$. Take α = bcegh. Since row b has dots in columns (9) and (10), row c in (11) and (12), row g in (14) and row h in (15), (16) and (17), the following alterms must be multiplied together: $(k \vee n)$, (m), $(i \vee o)$, (m), $(j \vee \ell \vee n \vee o)$, (i), $(j \vee m)$ and (k). Multiplying these alterms together yields $ijkm \vee ik\ell m \vee ikmn \vee ikmo$. Concatenating these terms with α yields bceghijkm, bceghikℓm, bceghikmn and bceghikmo which are terms in the irredundant disjunctive form representing the solution to Table 2.2. A more detailed description of this approach is given below.

METHOD 6:

STEP 1: Apply ORDERING 1 to the alterms formed from the left columns and then multiply them together in order from left to right (in the same manner as METHOD 1 multiplies alterms) to get a partial product.

STEP 2: For each upper row, v, of the CC-table, form the alterm (without the complemented literal) for each column which has a dot in row v. Multiply these alterms together in order as in METHOD 1 to get a partial product.

STEP 3: For each term in the partial product formed in STEP 1, select the necessary partial products from STEP 2. If literal v appears in the term from STEP 1, then the partial product formed by multiplying together the alterms corresponding to the columns with dots in row v is necessary. These necessary partial products are then multiplied together to get a new partial product.

STEP 4: Each term in the partial product formed in STEP 1 is juxtaposed with every term of the corresponding partial product in STEP 3 to form the closed covers of the CC-table.

Applying METHOD 6 to Fig. 3.2.1 yields

STEP 1: $(1)(3)(4)(2)$

$= (a \vee d)(c \vee d \vee e)(a \vee c)(b \vee c \vee e)$

$= (ac \vee d \vee ae)(a \vee c)(b \vee c \vee e)$

$= (ac \vee ad \vee ae \vee cd)(b \vee c \vee e)$

$= (abd \vee ac \vee cd \vee ae)$

STEP 2: row b: (f)

row c: (fg)

row e: (f $\vee$ g)

STEP 3: abd: (f)

ac: (fg)

cd: (fg)

ae: (f $\vee$ g)

STEP 4: (abdf $\vee$ acfg $\vee$ cdfg $\vee$ $\underline{\text{aef}}$ $\vee$ $\underline{\text{aeg}}$)

In the program which implements METHODS 1 through 6 a pair of computer words is used to represent each alterm formed from the CC-table. In the first word of the pair, the j th bit from the right is set to 1 if the corresponding column of the CC-table has an × in row j. Likewise, if the column has a dot in row i, then the i th bit from the right of the second word of the pair is set to 1. Since the leftmost bit of each word is used as a flag bit, the use of the above representation means that the program connot solve CC-tables which have more than 31 rows since the length of a computer word is 32 bits for the machines used. This restriction is valid for METHOD 1 through METHOD 6. However, in METHOD 6 the alterms corresponding to the cover columns and those corresponding to the closure columns are multiplied together separately. Furthermore, the alterms for the closure columns do not contain the complemented literals which correspond to the dots in the CC-table. This suggests the use of a new storage representation. A pair of computer words is still used to represent each column of the CC-table. For a cover column the first word of the pair has a 1 in the j th bit from the right if the column is covered by row j. All bits in the second word are set to 0. For a closure column

the first word has a 1 in the j th bit from the right if the column is covered by row M + j where M is the number of upper rows. The second word of the pair has a 1 in the k th bit from the right if the column is implied by row k. Only the first words of the pairs are used as the alterms to be multiplied together. The second words are used to determine which alterms are to be multiplied together according to the algorithm of METHOD 6. This storage representation allows CC-tables with a maximum of 31 upper rows and a maximum of 31 lower rows to be solved without increasing the the number of computer words necessary to represent the CC-table. METHODS 7 and 8 employ this storage representation along with the algorithm of METHOD 6.

METHOD 7: This is just METHOD 6 with the following step replacing STEP 1.

STEP 1: Apply ORDERING 1 to the alterms formed from the left columns and then multiply them together pairwise in the same manner as METHOD 5.

METHOD 8: This is just METHOD 7 except that ORDERING 2 is used in STEP 1.

The application of METHODS 7 and 8 to Fig. 3.2.1 is shown below.

METHOD 7:

STEP 1: $(1)(3)(4)(2)$

$= (a \vee d)(c \vee d \vee e)(a \vee c)(b \vee c \vee e)$

$= (ac \vee d \vee ae)(ab \vee c \vee ae)$

$= (abd \vee ac \vee cd \vee ae)$

STEP 2: row b: (f)

row c: (fg)

row e: $(f \vee g)$

STEP 3: abd: (f)

ac : (fg)

cd : (fg)

ae : (f $\vee$ g)

STEP 4: (abdf $\vee$ acfg $\vee$ cdfg $\vee$ $\underline{\text{aef}}$ $\vee$ $\underline{\text{aeg}}$)

METHOD 8:

STEP 1: (2)(3)(1)(4)

= (b $\vee$ c $\vee$ e)(c $\vee$ d $\vee$ e)(a $\vee$ d)(a $\vee$ c)

= (c $\vee$ bd $\vee$ e)(a $\vee$ cd)

= (ac $\vee$ abd $\vee$ ae $\vee$ cd)

STEP 2: row b: (f)

row c: (fg)

row e: (f $\vee$ g)

STEP 3: ac : (fg)

abd: (f)

ae : (f $\vee$ g)

cd : (fg)

STEP 4: (acfg $\vee$ abdf $\vee$ $\underline{\text{aef}}$ $\vee$ $\underline{\text{aeg}}$ $\vee$ cdfg)

3.4. Additional Remarks.

The calculations in Sections 3.2 and 3.3 illustrate the differences among the various methods. For this particular example the computation for each method is different from those for the other methods. This is not true in general since for some CC-tables ORDERING 1 and/or ORDERING 2 may not cause any alterms to be interchanged or ORDERING 1 and ORDERING 2 may be identical. At first glance it might appear that the final expression for METHOD 1 through METHOD 5 is not the same as the final expression for METHODS 6, 7, and 8. However, it must be remembered that only the uncomplemented variables are considered in finding the solutions from the final expression. If the uncomplemented variables are removed from the final expression for METHODS 1 through 5 and if terms which satisfy the realtion $x \vee xy = x$ are then removed, the result is the final expression for METHODS 6, 7, and 8. Thus all of the final expressions are consistant. Of course, all of the methods give the same solutions.

CHAPTER 4

COMPUTER EXPERIMENTS

Table 4.1 and Table 4.2 show the results of incorporating the methods described in Chapter 3 into a computer program and running some four and five variable functions on the IBM 360/75 at the University of Illinois. The output column in the truth table for each function is expressed in a hexadecimal number in these tables. Also the number of columns in the CC-table is shown. The purpose of these experiments was to determine what effect the various methods would have on the computation time required to solve the CC-table. The numbers shown in the tables under each method are the computation times (in seconds) which were required to solve the CC-table for the corresponding functions. Computer system subroutines were used to provide the computation times. All of the FORTRAN subroutines in the program were compiled in FORTRAN G.

Functions 1 through 5 of Table 4.1 were included to determine the effects of METHODS 2 through 8 on functions for which METHOD 1 requires a very short calculation time. The results in Table 4.1 indicate some fluctuations in the computation times for these five functions for some of the methods. However, no significant increases or decreases in the computation times are apparent for these five functions. Functions 6 through 12 have CC-tables which require much longer computation times for solution by METHOD 1. Table 4.1 shows that METHODS 2, 3, and 5 result in fairly significant (45% or better) reductions in the computation time for all of these functions except 10 and 11 while METHOD 4 shows little or no improvement for functions with a larger number of columns. Both functions

Functional (Hexadecimal)	No of Cols.	METHOD 1	METHOD 2	METHOD 3	METHOD 3'	METHOD 4	METHOD 5	METHOD 6	METHOD 7	METHOD 8
1. 0016	5	.00	.00	.00	.00	.00	.08	.00	.07	.10
2. 0998	9	.02	.04	.02	.03	.03	.15	.10	.10	.14
3. 879B	13	.30	.08	.08	.12	.12	.30	.11	.22	.19
4. 0118	15	.20	.15	.16	.15	.12	.28	.30	.32	.37
5. 19E3	17	.34	.13	.23	.11	.10	.30	.12	.22	.22
6. 4FA2 95F6	21	42.78	21.03	16.00	8.37	20.12	10.25	1.20	1.25	1.34
7. 98E2	23	8.87	4.85	5.47	2.62	6.32	1.79	.27	.38	.40
8. A6CD DF18	31	265.97	62.21	64.37	39.65	174.09	60.44	2.03	1.88	1.97
9. 8368	32	274.00	95.50	76.31	83.10	330.69	70.19	1.31	1.47	1.29
10. 2884	37	>600.00*	>600.00*	>600.00*	>600.00*	>600.00*	‡	9.62	9.79	17.49
11. FF68 A1F3	53	>600.00*	>600.00*	>600.00*	>600.00*	>600.00*	‡	4.31	3.61	3.48
12. FEE8	56	476.19	107.14	64.86	52.72	538.90	27.77	1.18	1.15	1.16

*The time assigned to this function was exhausted before the CC-table was solved.

‡The computation was stopped by the program because an array of dimension 40000 overflowed.

Table 4.1. The computation time (in seconds) required to solve the CC-table for some four and five variable functions.

Functional (Hexadecimal)	Number of Colums			Number of Upper Rows	Number of Lower Rows	METHOD 1	METHOD 6	METHOD 7	METHOD 8
	Left	Right	Total						
1. 4FA2 95F6	12	9	21	16	5	42.78	1.20	1.25	1.34
2. A6CD DF18	10	21	31	18	6	265.97	2.03	1.88	1.97
3. FF68 A1F3	8	45	53	17	14	>600.00*	4.31	3.61	3.48
4. 1EE6 5240	15	98	113	34	12	°	°	°	°
5. 9E63 BE7F	6	11	17	6	9	.05	.10	.19	.16
6. 0A88 8103	18	97	115	53	21	°	°	°	°
7. 49F3 63CD	11	29	40	16	15	‡	16.23	13.53	13.03
8. 8B58 09F0	17	40	57	33	14	°	°	°	°
9. BFD6 C6DA	9	27	36	15	6	5.33	.72	.93	.78
10. C6E7 103E	10	6	16	17	3	1.33	.63	.66	.58

*The time assigned to this function was exhausted before the CC-table was solved.

‡The computation was stopped by the program because an array of dimension 40000 overflowed.

°The CC-table has more rows than the program can handle.

Table 4.2. Statistics on the size of the CC-table and the time (in seconds) required to solve the CC-table for some five variable functions.

Functional (Hexadecimal)	Number of Columns			Number of Upper Rows	Number of Lower Rows	METHOD 1	METHOD 6	METHOD 7	METHOD 8
	Left	Right	Total						
11. 7A87 1D71	11	31	42	16	19	°	°	64.45	58.98
12. AE47 BF9F	6	15	21	9	7	.45	.20	.31	.21
13. AB95 69A2	12	19	31	21	9	>600.00*	5.92	5.13	5.16
14. 88B7 49B4	10	8	18	15	4	1.98	.46	.63	.58
15. 49C1 3031	13	65	78	36	18	°	°	°	°
16. 96B0 36A5	10	33	43	17	20	°	°	122.40	118.43
17. 686F 271F	8	25	33	14	14	‡	11.99	9.98	9.33
18. 5B23 D763	9	26	35	16	12	>600.00*	2.15	2.02	1.94
19. 858E C1CB	7	15	22	10	9	6.52	.21	.30	.25
20. D542 DA86	11	37	48	24	11	°	°	34.18	57.84

*The time assigned to this function was exhausted before the CC-table was solved.

‡The computation was stopped by the program because an array of dimension 40000 overflowed.

°The CC-table has more rows than the program can handle.

Table 4.2. Statistics on the size of the CC-table and the time (in seconds) required to solve the CC-table for some five variable functions (continued).

Functional (Hexadecimal)	Number of columns			Number of Upper Rows	Number of Lower Rows	METHOD 1	METHOD 6	METHOD 7	METHOD 8
	Left	Right	Total						
21. 11E3 52C2	10	50	60	22	9	>600.00*	4.80	4.10	3.58
22. 960D 65C8	12	41	53	25	17	°	°	‡	‡
23. 1D4C 022D	11	25	36	16	8	6.22	.70	.68	.78
24. A68E 9866	12	29	41	20	14	°	°	16.09	15.93
25. A8AA 5CE8	8	96	104	29	13	°	°	32.28	61.91
26. B42A 97A8	11	45	56	20	15	°	°	17.32	17.00
27. D58A 9F11	8	33	41	18	25	°	°	‡	‡
28. 4D16 B414	12	31	43	23	10	°	°	29.27	18.46
29. DA18 2E51	15	51	66	24	19	°	°	‡	‡
30. 3957 22CD	9	23	32	14	16	>600.00*	6.54	4.96	4.79

*The time assigned to this function was exhausted before the CC-table was solved.

‡The computation was stopped by the program because an array of dimension 4000 overflowed.

°The CC-table has more rows than the program can handle.

Table 4.2. Statistics on the size of the CC-table and the time (in seconds) required to solve the CC-table for some five variable functions (continued).

10 and 11 have CC-tables with 31 rows (the maximum allowed for METHODS 1 through 5) which at least partially explains why METHODS 1 through 5 were not able to solve the CC-tables for these functions without using an excessive amont of computation time and/or core storage. A comparison of the results of METHOD 3 to those of METHOD 3' was used as a basis for incorporating the changes described in the discussion of METHOD 3' into METHODS 4 through 8.

Somewhat surprisingly, it is the calculation procedure of METHOD 6 (the alterms corresponding to the cover columns are multiplied together first and then subsets of the alterms corresponding to closure columns are multiplied together as necessitated by the dots in the CC-table) which causes very significant (97% or better for functions 6 through 12 of Table 4.1) reductions in the computation time. METHODS 7 and 8 are basically just extensions of METHOD 6 as discussed previously. The pairwise multiplication of METHOD 5 is used to multiply the alterms corresponding to the cover columns in METHODS 7 and 8 on the basis of the results in Table 4.1. METHODS 7 and 8 give results comparable to METHOD 6. The differences between the results for METHOD 7 and METHOD 8 can be attributed to the differences between ORDERING 1 and ORDERING 2.

Table 4.1 clearly establishes the superiority of METHODS 6, 7 and 8. The thirty functions of five variables shown in Table 4.2 were used to further test the effectiveness of METHODS 6, 7 and 8. Table 4.2 shows not only the computation time (in seconds) required to solve the CC-table of each function for the various methods but also the size of the CC-table for each function. The results in Table 4.2 further substantiate the conclusions drawn from Table 4.1. That is, METHODS 6, 7 and 8 are much faster than METHOD 1; for a given function the computation times for METHODS 6, 7 and 8 are similar (except for two or three functions); and METHODS 6, 7 and 8 can quickly solve some functions which can not be solved by METHOD 1 without

using an excessive amount of computation time and/or core storage. Table 4.2 also contains several functions which have too many rows to be solved by METHOD 6 but which can be solved by METHODS 7 and 8 as well as several functions which have too many rows to be solved by any of the methods. Furthermore, there are several functions which can not be solved even by METHODS 7 and 8 without using an excessive amount of core storage and probably a long computation time. As an illustration consider function 29 of Table 4.2 and the algorithm outlined in the discussion of METHOD 6 given previously. In STEP 1 multiplying together the 15 alterms corresponding to cover columns results in a partial product with 1846 terms. For the first 250 of these terms, STEP 3 yields partial products with an average of 66 terms. Assuming this average is reasonably accurate for all 1846 terms in the partial product in STEP 1, STEP 4 would result in more than 121,000 closed covers for the CC-table

In summary, the results of Tables 4.1 and 4.2 show that significant reductions in the computation time can be achieved. Rearranging the alterms by applying an ordering algorithm is usually helpful. More sophisiticated orderings may result in further reductions of the computation time if applying the ordering to the alterms does not require much computation time. ORDERING 2 is somewhat more complicated than ORDERING 1, but a comparison of the results by METHOD 7 to those by METHOD 8 show that the two orderings give similar results. For this reason orderings more sophisticated than ORDERING 2 were not attempted. Using the algorithm of METHODS 6, 7 and 8 results in the greatest decrease in the computation time which appears to be a complicated function of many variables including the number of left columns, the number of right columns, the number of upper rows, the number of lower rows, and the density of X's in the cc-table. Unfortunately, there are functions of only five variables which have CC-tables that are too large

to be solved even by METHODS 7 and 8. Further improvements to the program may allow some of these functions to be solved, but it seems likely that there would still exist functions whose CC-tables would be too large to be solved by computer.

ACKNOWLEDGEMENT

The authors wish to sincerely thank Dr. Gimpel for his cooperation in sending us a copy of his computer program which implements his algorithm with METHOD 1. Without his kind consideration the experiments and results given in this paper would not have been possible.

REFERENCES

[1] James F. Gimpel, "The Minimization of TANT Networks," IEEE Transactions on Electronic Computers, Vol. EC-16, pp. 18-38, February, 1967.

[2] A. Grasselli and F. Luccio, "A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Machines," IEEE Transactions on Electronic Computers, Vol. EC-14, pp. 350-359, June, 1965.

[3] E.J. McCluskey, "Introduction to the theory of switching circuits," McGraw-Hill Book Co., 155 pages, 1965.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-720	2.	3. Recipient's Accession No.

4. Title and Subtitle
ALTERNATIVE METHODS FOR SOLVING THE CC-TABLE IN GIMPEL'S ALGORITHM FOR SYNTHESIZING OPTIMAL THREE LEVEL NAND NETWORKS

5. Report Date
April 1975

6.

7. Author(s)
Keith R. Hohulin and Saburo Muroga

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
NSF GJ-40221

12. Sponsoring Organization Name and Address
National Science Foundation
1800 G Street, N.W.
Washington, D.C. 20550

13. Type of Report & Period Covered
Technical

14.

15. Supplementary Notes

16. Abstracts

In the February 1967 issue of IEEE Transactions on Computers, Gimpel presents an algorithm for finding the TANT networks with the minimum number of gates for a given switching function. The algorithm involves the formation and solution of a cover and closure table (CC-table) which are major bottlenecks in computer processing of the algorithm. This paper presents several alternative methods for solving the CC-table in Gimpel's algorithm, which are suitable for computer processing. It also presents the results of implementing these methods into computer programs and using them to find the TANT networks with the minimum number of gates for some randomly selected switching functions.

17. Key Words and Document Analysis. 17a. Descriptors

Logical design, Gimpel, NAND gates, TANT networks, cover and closure table, optimal networks

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	UNCLASSIFIED	29
	20. Security Class (This Page) UNCLASSIFIED	22. Price

INSTRUCTIONS FOR COMPLETING FORM NTIS-35 (10-70) (Bibliographic Data Sheet based on COSATI Guidelines to Format Standards for Scientific and Technical Reports Prepared by or for the Federal Government, PB-180 600).

1. **Report Number.** Each report shall carry a unique alphanumeric designation. Select one of the following types: (a) alphanumeric designation provided by the sponsoring agency, e.g., **FAA-RD-68-09;** or, if none has been assigned, (b) alphanumeric designation established by the performing organization e.g., **FASEB-NS-87;** or, if none has been established, (c) alphanumeric designation derived from contract or grant number, e.g., **PH-43-64-932-4.**

2. Leave blank.

3. **Recipient's Accession Number.** Reserved for use by each report recipient.

4. **Title and Subtitle.** Title should indicate clearly and briefly the subject coverage of the report, and be displayed prominently. Set subtitle, if used, in smaller type or otherwise subordinate it to main title. When a report is prepared in more than one volume, repeat the primary title, add volume number and include subtitle for the specific volume.

5. **Report Date.** Each report shall carry a date indicating at least month and year. Indicate the basis on which it was selected (e.g., date of issue, date of approval, date or preparation.

6. **Performing Organization Code.** Leave blank.

7. **Author(s).** Give name(s) in conventional order (e.g., John R. Doe, or J.Robert Doe). List author's affiliation if it differs from the performing organization.

8. **Performing Organization Report Number.** Insert if performing organization wishes to assign this number.

9. **Performing Organization Name and Address.** Give name, street, city, state, and zip code. List no more than two levels of an organizational hierarchy. Display the name of the organization exactly as it should appear in Government indexes such as **USGRDR-I.**

10. **Project/Task/Work Unit Number.** Use the project, task and work unit numbers under which the report was prepared.

11. **Contract/Grant Number.** Insert contract or grant number under which report was prepared.

12. **Sponsoring Agency Name and Address.** Include zip code.

13. **Type of Report and Period Covered.** Indicate interim, final, etc., and, if applicable, dates covered.

14. **Sponsoring Agency Code.** Leave blank.

15. **Supplementary Notes.** Enter information not included elsewhere but useful, such as: Prepared in cooperation with . . . Translation of . . . Presented at conference of . . . To be published in . . . Supersedes . . . Supplements . . .

16. **Abstract.** Include a brief (200 words or less) factual summary of the most significant information contained in the report. If the report contains a significant bibliography or literature survey, mention it here.

17. **Key Words and Document Analysis. (a). Descriptors.** Select from the Thesaurus of Engineering and Scientific Terms the proper authorized terms that identify the major concept of the research and are sufficiently specific and precise to be used as index entries for cataloging.
(b). Identifiers and Open-Ended Terms. Use identifiers for project names, code names, equipment designators, etc. Use open-ended terms written in descriptor form for those subjects for which no descriptor exists.
(c). COSATI Field/Group. Field and Group assignments are to be taken from the 1965 COSATI Subject Category List. Since the majority of documents are multidisciplinary in nature, the primary Field/Group assignment(s) will be the specific discipline, area of human endeavor, or type of physical object. The application(s) will be cross-referenced with secondary Field/Group assignments that will follow the primary posting(s).

18. **Distribution Statement.** Denote releasability to the public or limitation for reasons other than security for example "Release unlimited". Cite any availability to the public, with address and price.

19 & 20. **Security Classification.** Do not submit classified reports to the National Technical Information Service.

21. **Number of Pages.** Insert the total number of pages, including this one and unnumbered pages, but excluding distribution list, if any.

22. **Price.** Insert the price set by the National Technical Information Service or the Government Printing Office, if known.